Consider two following representations of a non-negative integer:

1. A simple decimal integer, constructed of a non-empty sequence of digits from 0 to 9;
2. An integer with at least one digit in a base from 2 to 16 (inclusive), enclosed between # characters, and preceded by the base, which can only be a number between 2 and 16 in the first representation. For digits from 10 to 15 characters a, b, ..., f and A, B, ..., F are used.

Additionally, both representations may contain *underscore* (\_) characters; they are used only as separators for improving legibility of numbers and can be ignored while processing a number.

Your task is to determine whether the given string is a valid integer representation.

*Note: this is how integer numbers are represented in the programming language Ada.*

**Example**

* For line = "123\_456\_789", the output should be  
  adaNumber(line) = true;
* For line = "16#123abc#", the output should be  
  adaNumber(line) = true;
* For line = "10#123abc#", the output should be  
  adaNumber(line) = false;
* For line = "10#10#123ABC#", the output should be  
  adaNumber(line) = false;
* For line = "10#0#", the output should be  
  adaNumber(line) = true;
* For line = "10##", the output should be  
  adaNumber(line) = false.

**Input/Output**

* **[time limit] 3000ms (cs)**
* **[input] string line**

A non-empty string.

*Constraints:*  
2 ≤ line.length ≤ 30.

* **[output] boolean**

true if line is a valid integer representation, false otherwise.

<https://codefights.com/arcade/code-arcade/well-of-integration/Ghe6HWhFft8h6fR49>

<https://codefightssolver.wordpress.com/2016/11/page/7/>

bool adaNumber(string line)

{

bool atLeastOneDigit = false;

if (line[line.Length - 1] == '#')

{

int i = 0;

int baseChar = 0;

while (line[i] != '#' && baseChar <= 16)

{

if (line[i] != '\_')

{

if ('0' <= line[i] && line[i] <= '9')

{

baseChar = baseChar \* 10 + (line[i] - '0');

}

else

{

return false;

}

}

i++;

}

if (baseChar < 2 || baseChar > 16)

{

return false;

}

i++;

while (i < line.Length - 1)

{

if (line[i] != '\_')

{

var digit = -1;

if ('a' <= line[i] && line[i] <= 'f')

{

digit = line[i] - 'a' + 10;

}

if ('A' <= line[i] && line[i] <= 'F')

{

digit = line[i] - 'A' + 10;

}

if ('0' <= line[i] && line[i] <= '9')

{

digit = line[i] - '0';

}

if (0 <= digit && digit < baseChar)

{

atLeastOneDigit = true;

}

else

{

return false;

}

}

i++;

}

}

else

{

foreach (char i in line)

{

if (i != '\_')

{

if ('0' <= i && i <= '9')

{

atLeastOneDigit = true;

}

else

return false;

}

}

}

return atLeastOneDigit;

}